Lab 15: Controller Benchmarks

Time Estimate: 1.5 hours

Training Objective: To investigate the implementation of the watermover control in the Regional Simulation Model.

Watermover control is obtained directly through Hydrologic Simulation Engine (HSE) watermover controllers and indirectly through the use of assessors in the Management Simulation Engine (MSE). This lab explores the implementation of user-defined controllers, which have been implemented in the HSE, and the implementation of the WMM_assessors to control watermovers.

NOTE:

For ease of navigation, you may wish to set an environment variable to the directory where you install the RSM code using the syntax

```
setenv RSM <path>
```

For SFWMD modelers, the path you should use for the NAS is:

```
/nw/oomdata_ws/nw/oom/sfrms/workdirs/<username>/trunk
```

```
setenv RSM
/nw/oomdata_ws/nw/oom/sfrms/workdirs/<username>/trunk
```

Once you have set the RSM environment variable to your trunk path, you can use $RSM in any path statement, such as:

```
cd $RSM/benchmarks
```

Training files are currently located in the following directories:

```
INTERNAL_TRAINING
|
|____data
|       |__geographic
|       |__C111
|       |__rain+et
|       |__glades_lesca
|       |__losa_eaa
|       |__BBCW
|
|____trunk
|       |___benchmarks
|       |___hpmbud
|       |___budtool (obsolete?)
|
|____labs
```

Files for this lab are located in the **`labs/lab15_BM1`** directory. Additional materials in the directory include:

Activity 15.1 User-Specified Controllers

Overview

Activity 15.1 includes 2 exercises:

- **Exercise 15.1.1.** Modify <userctrl> in Benchmark 45.
- **Exercise 15.1.2.** Modify <userctrl> for SR29 in Glades-LECSA RSM

The Regional Simulation Model (RSM) has the capability of implementing user-defined controllers, which are described in Benchmark 45. There are four examples in BM45. These test the use of controllers and supervisors written in either the C++ or the C language. We will investigate the use of controllers in the C++ environment.

Exercise 15.1.1 Modify <userctrl> in Benchmark 45

1. In the trunk folder, copy the **`RSM/trunk/benchmarks/BM45`** directory to new directory: **`BM45a`**
   - **`cp -r BM45 BM45a`**
2. Run test.script to ensure **`BM45a`** is working correctly
3. Run the simple controller (**`run3x3.xml`**) from the RSM Graphical User Interface (RSM GUI)
4. Observe the results in **`t3x3out.dss`** using the **`HecDssVue`** utility
5. Select canal S01 head, WM1 control and WM1 flow. Keep the graph

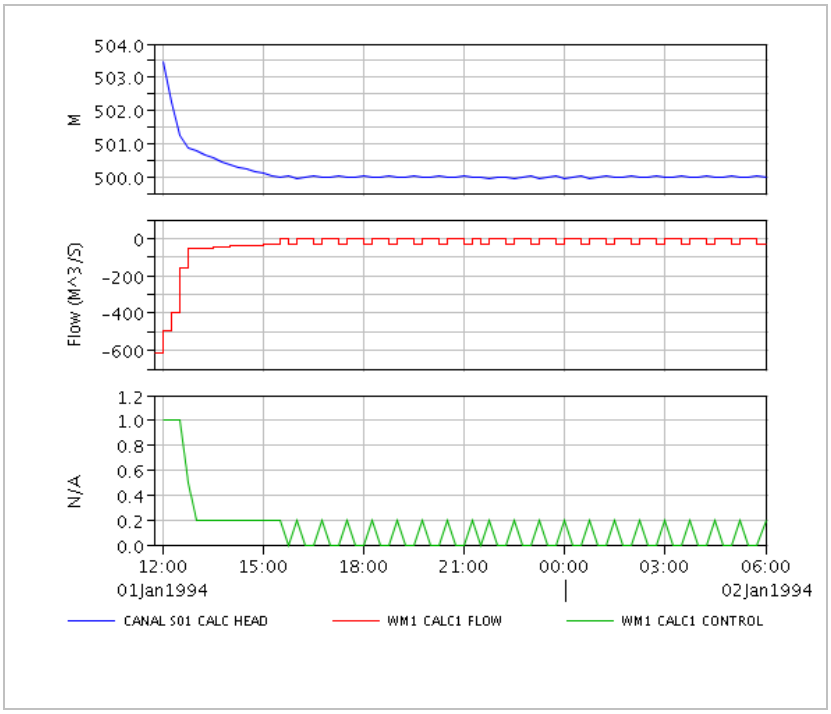


Figure 15.1 Time series of stage, flow and control-setting for the user_ctrl

6. Edit the run file (**`run3x3.xml`**) and turn off the controller cid="**`101`**"

```
<!-- Watermovers to be controlled -->
<watermovers>
  <!-- discharge from canal segment 1 -->
  <hq_relation wmID="1" id="1" label="">
     <hq>
      0.0    0.0
      495.0  0.0
      499.0 -50.0
      500.0 -150.0
      501.0 -300.0
      510.0 -1000.0
     </hq>
  </hq_relation>
  <!-- inflow into canal segment 4 -->
  <hq_relation wmID="2" id="4" label="">
     <hq>
      0.0    0.0
      490.0  1000.0
      495.0  500.0
      499.0  250.0
      500.0  150.0
      501.0  50.0
      510.0  0.0
     </hq>
  </hq_relation>
</watermovers>

<controller id="1">
  <!-- Controller for discharge from segment 1 -->
  <userctrl cid="101" label="Segment 1 Ctrl " wmID="1" control="on"
   libType="C" module="./User_C.so" func="Segment1_Control" ctrlMin="0.0" ctrlMax="1.0" >
     <varIn name="Segment1"><segmentmonitor id="1" attr="head"/></varIn>
     <varIn name="Segment4"><segmentmonitor id="4" attr="head"/></varIn>
  </userctrl>
</controller>
<controller id="2">
  <!-- Controller for pumping into segment 4 -->
  <userctrl cid="102" label="Segment 4 Ctrl" wmID="2" control="on"
   libType="C" module="./User_C.so" func="Segment4_Control" ctrlMin="0.0" ctrlMax="1.0" >
     <varIn name="Segment1"><segmentmonitor id="1" attr="head"/></varIn>
     <varIn name="Segment4"><segmentmonitor id="4" attr="head"/></varIn>
  </userctrl>
</controller>
```

Figure 15.2 The watermovers and control functions used with user-specified controllers

7. Rerun model, display results and compare to the previous results
   - How does the stage behave?
8. Edit the **`userctrl.cc`** file and change the lower control value from 500 to 499.5

   The controller behavior is described in a C++ that sets the control variable. In this case the variables "**`Segment1_Control`**" and "**`Segment4_Control`**" are set in the **`userctrl.cc`** file (**Fig. 15.2**). The user-controller is implemented using the **`<controller>`** block in the file (**Fig 15.3**)
9. Recompile the user-control library by executing the makefile script using the "make" command
10. Rerun the model and compare current results to previous results

11. Change the output gain "**`controlOut`**" from 1.0 to 0.60 and run the model
    - Compare the results to previous results

12. Change the **`controlOut`** to ***6.0*** and observe the results

```
//-----------------------------------------------------------------------
// Function Segment1_Control for control of watermover into segment 1
//
extern "C" double Segment1_Control( map<string, InputState*> *lpInputStateMap ) {

  string func = "Segment1_Control";
  double controlOut = 0.;

  double segment1Head = GetVarIn( func, "Segment1", lpInputStateMap );
  // These are not used, but illustrate how to get values from XML input
  double xmlScalarVal = XMLScalarValue( func, "xmlScalar",         lpInputStateMap );
  double xmlVectorVal = XMLVectorValue( func, "xmlVector",  2,     lpInputStateMap );
  double xmlMatrixVal = XMLMatrixValue( func, "xmlMatrix",  2, 3, lpInputStateMap );

  // Provide control function based on input state variable
  if ( segment1Head > 502. )
    controlOut = 1.;
  else if ( segment1Head > 501. )
    controlOut = 0.5;
  else if ( segment1Head > 500. )
    controlOut = 0.2;
  else
    controlOut = 0.;

  return controlOut;
}
```

Figure 15.3 The userctrl.cc file used to set rules for controlling structure

Activity 15.2 WMM-Assessor (Benchmark 63b)

Overview

Activity 15.2 does. This activity includes 1 exercises:

- **Exercise 15.2.1.** Run WMM-Assessors in BM 63b.

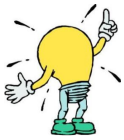
Exercise15.2.1 Run WMM-Assessors in BM 63b

User-defined controllers can be used for specific structures but are rarely used in the current versions of the subregional models. Effective for controlling the behavior of individual structures, user-defined controllers have proven expensive to scale-up to the subregional or regional implementations.

In this exercise you will find the user-defined controllers and describe how they are used.

1. In the **`/RSM/data/glades_lecsa`** directory, find the **`glades-controllers.xml`**

HINT



Find the `<controller>` block in the `run_81.xml` file.

2. Find the controllers for the SR29 structures:
   - What are the inputs to the controller?
   - What is the file with the control variables?
   - What does the control rule in SR29_seasonalctrl.cc do?
   - What watermovers does this control?
   - What does the control behavior look like?
   - What are the flow values for the structures?
3. Run the model and graph the flow data for the SR29 structures.

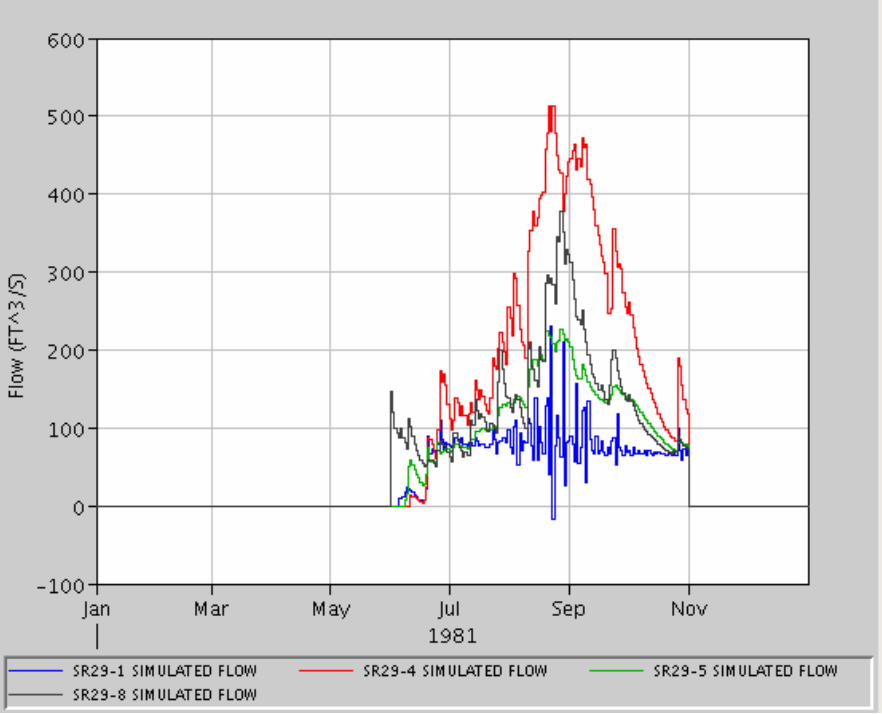
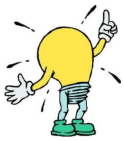


Figure 15.4 Simulated discharge at the SR29 weirs

4. Edit the **`SR29_seasonalctrl.cc`** and change the dryseason control to *0.2*.
5. Recompile the control file using the instructions in **`SR29_seasonalctrl.cc.`**

HINT



The compilation instructions are at the top of the file.

6. Run the model using RSM GUI and compare the results to the previous run.

Activity 15.2 WMM_Assessor (Benchmark 63b)

Overview

Activity 15.2 does. This activity includes 1exercise:

- **Exercise 15.2.1.** Run WMM-Assessors in BM 63b.

The WMM_Assessor uses assessors to determine the amount of water from the Hydrologic Simulation Engine (HSE) that needs to be moved to or from a water control unit to meet the water supply (ws) needs or flood control (fc) requirements for each basin. The user-entered mse_units and mse_nodes are used to construct the <mse_network>. The wmm_assessor <WMM> determines the required management of the structures <mse_struc> from "off" to full "on" at each managed structure, based on the requirements for each structure.

This benchmark was designed to test the behavior of the WMM_Assessor. In order to understand the structure of the mse_network, map out the network manually:

1. In the benchmark folder (**`/RSM/trunk/benchmarks`**) copy Benchmark 63b to the BM63b1 directory. All work for this activity will be completed in the **`lab16_mse/BM63b`** directory.
   - **`cp –r ./RSM/trunk/benchmarks/BM63b ./RSM/trunk/benchmarks/BM63b1`**
2. List the **`mse_network.xml`** file.
3. Create a diagram of the mse_network connecting the mse_units using the mse_nodes.
   - List the water supply maintenance levels next to each **`<mse_unit>`**
   - List the maximum discharge capacity next to each **`<mse_node>`** (structure)
   - List the flood control level for each reach. (Look at **`<mse_node>`** openings)
4. Construct a plot that shows the discharge behavior of **`<mseStruc>`** **S9** and **S8**.

   The flow is calculated as follows:

   $Q = \text{dischar}\ \Delta h^{a}$

   - The default for a is *0.5* (if not specified)
5. Add the **`<runDescriptor>`** element to the **control** section.
   - **`hseMaxIteration`** = "**`144`**"
   - **`runDescriptor`** = "**`forest`**"
6. Run Benchmark 63b using the RSM GUI.

7. Observe the following data using HecDssVue from the RSM GUI:
   - Cell heads (cellheads.dss) Find the cell locations and WCU using the Groundwater Modeling System? (GMS) graphic for mesh and network, **Fig. 15.1** in the Answer-lab15 file
     - What do these values indicate?
   - Segment heads (segmentheads.dss)
     - Which WCUs become flooded?
   - Structure flows (structureflows.dss)
   - Nodeflows
     - Plot the QWS and QFC for a couple of nodes. What do you see?
   - Lake
   - Bcflows (These are the flows into or out of the domain. This may be important where you are trying to produce net outflow, balance or net inflow.)
   - Iterations

   The Management Simulation Engine (MSE) sets the flows at the structures and these are imposed on the Hydrologic Simulation Engine (HSE) matrix solution. The Regional Simulation Model iterates the HSE solution until the HSE flows converge with the MSE flows. This plot shows the number of iterations.
8. Replace the marsh Hydrologic Process Module (HPM) with an irrigated land HPM.
9. Change the **HPM** from **<layer1nsm>** (forest) to **<afsirs>** (turf) by substituting the following code (Fig. 15.5).
10. Change the runDescriptor in the **<control>** block to "turf".
11. Save the model as **wmm_assessors_b.xml.**

```
<hpModules>
  <indexed file="hpm.index">
    <hpmEntry id="1">
      <afsirs>
        <afcrop label="turf" id="15" j1="01-01" jn="12-31" depth1="12"
depth2="24">
          <kctbl>
             1.00 1.00 1.00 1.00 1.00 1.00
             1.00 1.00 1.00 1.00 1.00 1.00
          </kctbl>
          <awdtbl>
             0.50 0.50 0.50 0.50 0.50 0.50
             0.50 0.50 0.50 0.50 0.50 0.50
          </awdtbl>
        </afcrop>
        <afirr label="MULTIPLE SPRINKLER" wtd="2.5">
          <irrmeth id ="4" eff="0.85" arzi="1.0" exir="0.7"></irrmeth>
          <irrmgmt label = "DROUGHT" trigcode="2" value="100"></irrmgmt>
        </afirr>
        <afsoil label="SANDY LOAM SOILS" depth="96" minwc="0.09"
maxwc="0.15" cond="1">
        </afsoil>
      </afsirs>
    </hpmEntry>
  </indexed>
</hpModules>
```

Figure 15.5 Title

12. Run the model using the RSM GUI.
13. Observe the results compared to the previous run as “forest” by opening the following DSS files:
    - cellheads.dss
    - segmentheads.dss
    - nodeflows.dss
    - iterations.dss
14. Use HecDssVue and plot the “forest” time series with the “turf” time series.
    - How have the cell and segment heads changed from the forest HPMs?
    - How have the nodeflows changed?
    - How has the number of iterations changed?
15. Change the water management level in each WCU by adding a “reserve level” to the WCUs at the upper end of the regional system. (This will allow water to be delivered to the lower reaches to allow those WCUs to maintain higher water tables.)

16. In the **mse_network.xml** file add and replace the following lines to mse_unit name= ***"Reach#"***:

```
      <mse_unit name="Reach1">
         <hse_arcs> 100 101 102 103 104 105 106 107 118 119 120 121

         </hse_arcs>
         <inlet>  "S12" </inlet>
         <outlet> "S7"  </outlet>
         <outlet> "S9"  </outlet>
         <maintLevel name="reach1 maint"> <const value="4.5"/>
</maintLevel>
         <resLevel name="reach1 reserve"> <const value="3.5"/> </resLevel>
      </mse_unit>
```

Figure 15.6 Title

- Add to Reach1:

```
         <resLevel name="reach1 reserve"> <const value="3.5"/> </resLevel>
```

- Add to Reach2:

```
         <resLevel name="reach2 reserve"> <const value="3.0"/> </resLevel>
```

- Add to Reach3:

```
         <resLevel name="reach3 reserve"> <const value="2.75"/>
</resLevel>
```

17. Save the file as **mse_networkb.xml**.
18. Change **<mse_network>** in **wmm_assessorb.xml** to **xml="mse_networkb.xml".**
19. Change the **runDescriptor** in the **<control>** block to "**reserve**". Save the file.
20. Run **mse_network.xml** and observe the results:
    - cellheads.dss
    - segmentheads.dss
    - nodeflows.dss
      - How have the cell and segment heads changed?
      - How have the nodeflows changed?

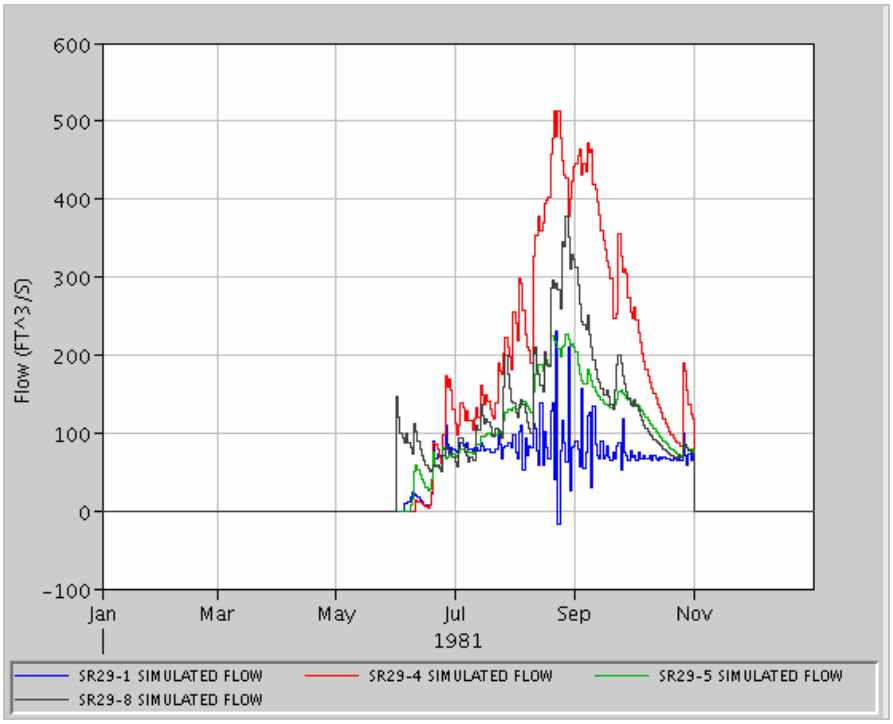


Figure 15.7 Title

Index

_Assessor (Benchmark 63b), 8
“forest” time series, 10
“reserve level”, 10
“reserve”, 11
“turf” time series, 10
<afsirs>, 9
<control>, 9, 11
<control> block, 11
<controller> block, 4, 6
<layer1nsm>, 9
<mse_network>, 8, 11
<mse_node>, 8
<mse_unit>, 8
<mseStruc>, 8
<runDescriptor>, 8
assessors, 8
balance, 9
Bcflows, 9
benchmark, 8
Benchmark 45, 3, 6
Benchmark 63b, 8
Benchmark 63b (BM63b), 8
benchmarks, 2
BM45, 3, 6
BM45a, 3
BM63b, 8
BM63b1, 8
budtool, 2
C++, 4, 6
canal S01 head, 3
Cell heads, 9
Cell heads (cellheads.dss), 9
cellheads.dss, 9, 10, 11
compilation instructions, 7
control, 8
control behavior, 6
control file, 7
control rule, 6
control variable, 4
control variables, 6
control watermovers, 1
controller, 6
controller behavior, 4
Controller Benchmarks, 1
controller cid=”101”, 4
controllers, 3, 6
controlling the behavior of individual structures, 6
controlOut, 5
current versions of the subregional models, 6
determine the amount of water from the Hydrologic Simulation Engine (HSE) that needs to be moved to or from a water control unit to meet the water supply (ws) needs or flood control (fc) requirements for each basin, 8
discharge behavior, 8
dryseason control, 7
environment variable, 2
executing the makefile script, 4
flood control (fc), 8
flood control level, 8
flow data, 6
flow values, 6
forest HPMs, 10
glades-controllers xml, 6
Glades-Lower East Cost Service Area, 6
Glades-Lower East Cost Service Area (LECSA), 6
Glades-Lower East Cost Service Area (LECSA) subregional model, 6
GMS, 9
Groundwater Modeling System, 9
Groundwater Modeling System? (GMS) graphic for mesh and network, 9
HecDssVue, 3, 9, 10
HecDssVue utility, 3
HPM, 9
hpmbud, 2
HSE, 1, 8, 9
HSE flows, 9
HSE solution, 9
hseMaxIteration, 8
Hydrologic Process Module, 9
Hydrologic Simulation Engine, 1, 8, 9
Hydrologic Simulation Engine (HSE), 1, 8
Hydrologic Simulation Engine (HSE) matrix solution, 9
Hydrologic Simulation Engine (HSE) watermover controllers, 1
Implement the user-controller, 3
implementation of the watermover control in the Regional Simulation Model, 1
implementation of the WMM_assessors to control watermovers, 1
implementation of user-defined controller in Glades-Lower East Cost Service Area (LECSA) subregional model, 6
implementation of user-defined controllers, 1
implementing user-defined controllers, 3, 6
inputs, 6
inputs to the controller, 6
irrigated land, 9
irrigated land HPM, 9
iterations, 9, 10
Iterations, 9
iterations.dss, 10
Lake, 9
LECSA, 6
lower control value, 4
make, 4

make" command, 4
makefile script, 4
managed structure, 8
management of the structures <mse_struc>, 8
Management Simulation Engine, 1, 9
Management Simulation Engine (MSE), 1, 9
map out the network by hand, 8
marsh, 9
marsh Hydrologic Process Module (HPM), 9
maximum discharge capacity, 8
MSE, 9
MSE flows, 9
mse_network, 8
mse_network.xml, 11
mse_network.xml file, 8
mse_nodes, 8
mse_units, 8
net inflow, 9
net outflow, 9
nodeflows, 10, 11
Nodeflows, 9
nodeflows.dss, 10, 11
note, 2
output gain "controlOut", 4
produce net outflow, 9
Q= dischar Δh^a, 8
QFC, 9
QWS, 9
rain, 2
Regional Simulation Model, 9
regional system, 10
RSM Graphical User Interface, 3
RSM Graphical User Interface (RSMGUI), 3
RSMGUI, 3, 7, 8, 9, 10
run file, 4
runDescriptor, 8, 9, 11
segment heads, 10, 11
Segment heads, 9
Segment1_Control, 4
Segment4_Control, 4
segmentheads.dss, 9, 10, 11
setenv, 2
simple controller, 3
Simulated discharge at the SR29 weirs, 7
SR29 structures, 6
SR29_seasonalctrl.cc, 6, 7
stage, 4
structure of the mse_network, 8
subregional model, 6
subregional models, 6
test the behavior of WMM-Assessor, 8
The userctrl.cc file used to set rules for controlling structure, 5
The watermovers and control functions used with User-specified controllers, 4
Time series of stage, flow and control-setting for the user_ctrl, 3
use of assessors, 1
use of assessors in the Management Simulation Engine (MSE) to control watermover, 1
use of controllers and supervisors written in either C++ or the C language, 3
use of the controllers in the C++ environment, 6
user-control library, 4
user-controller, 4
userctrl.cc file, 4, 5
user-defined controller, 6
User-defined controller Benchmark 45 (BM45), 3
user-defined controllers, 1, 3, 6
User-defined controllers, 6
Water Control Unit, 9
Water Control Units, 9
water management level, 10
water supply (ws), 8
water supply maintenance levels, 8
watermover control, 1
Watermover control, 1
watermovers, 6
WCU, 9, 10
WCUs, 10
We will be investigating the use of the controllers in the C++ environment, 3
WM1 control, 3
WM1 flow, 3
WMM_Assessor, 8
wmm_assessor <WMM>, 8
WMM_assessors, 1
WMM-Assessor, 8